Discourse representation meets constructive mathematics

Ahn, R.; Kolb, H.P.

Publication date:
1990

Document Version
Publisher's PDF, also known as Version of record

E1

[redacted]

ITK RESEARCH REPORT

INSTITUTE FOR LANGUAGE TECHNOLOGY AND ARTIFICIAL INTELLIGENCE

INSTITUUT VOOR TAAL- EN KENNISTECHNOLOGIE

ITK Research Report No. 16

April 4, 1990

Discourse Representation meets Constructive Mathematics

René Ahn
Hans-Peter Kolb

ISSN 0924-7807

Discourse Representation meets Constructive Mathematics

René Ahn **Hans-Peter Kolb**

ITK, Tilburg University
P.O.Box 90153
NL-5000 LE Tilburg
The Netherlands

April 3, 1990

Abstract

The principal aim of this paper is to point out a striking similarity between Discourse Representation Structures [7],[8],[9] and so called contexts in a class of formalisms known as Constructive Type Theories [2],[3]. These formalisms are based on the *propositions as types* concept which goes back to Curry [4]. The universally desirable features of CTT[1] include:

- CTT has a small, elegant and very powerful set of deduction rules.
- CTT is a higher order[2] formalism, i.e. we can quantify over arbritrary predicates and functions within CTT. As a result anaphoric references to such entities are also possible.
- Every proof in CTT is a λ-expression, that can be directly investigated and manipulated or reduced. For instance, β-reduction on these λ-expressions corresponds directly to cut elimination.
- Moreover, CTT is currently one of the mainstream research areas in theoretical computer science. Therefore there exists an impressive amount of metalogical knowledge about the formalism.

It will be argued that a DRS is in fact a first order approximation of a CTT context. Thus, it seems possible to combine the empirical advantages of DRT with the mathematical rigour as well as the deductive and descriptive power of CTT.

After an informal introduction to CTT in general, which aims at getting across the basic intuitions relevant to such formalisms, we will select a particular system and present its formal definition along conventional lines. Unfortunately, the resulting system is hard to grasp, partly due to inherent difficulties, but largely for syntactic reasons, and because the presentation does not lend itself easily to a comparasion with familiar systems like, say, classical first order logic

[1] A related system better known in linguistics is that of Martin Löf. (cf. [12], [13])

[2] ...in fact, it contains any n'th order logic.

(CFOL). By a change in representation based on [1], [5] we succeed in isolating, so to speak, the first order part of the theory, and end up with a *natural deduction*-like system that is both easier to understand and use.

In the second part we present a mapping to translate any Discourse Representation Structure into a (first order and uniformally typed) fragment of a CTT context. The modeltheoretic truth conditions of DRT translate in a natural way into satisfiability conditions on CTT-fragments (corresponding to DRSes) with respect to a given context. Classical semantics is obtained by checking satisfiability with respect to the complete context representing the knowledge of the omniscient infallible being Ω. But since, as a rule, CTT contexts represent only partial knowledge of a world, CTT enables us to make sense of situations which crucially involve partial and possibly incorrect knowledge, such as communication, as well. The formalism also provides us with handles to cope with belief revision and truth maintenance. This, we believe, is a highly desirable feature for any realistic linguistic semantic theory.

1 CTT

1.1 General Ideas

Type theories specify a relation between objects and types, often denoted by the infix operator ':'. This relation depends on a *context*, which contains all the relevant information about primitive and prefabricated objects. The objects and types are denoted by expressions, and the ':' relation is specified recursively by a collection of inference rules. More precisely, one infers formulae of the form $C \vdash [E : T]$, where C is a context, E is an expression denoting an object and T is an expression denoting its type. It is a typical feature of CTT that an expression can, on one hand, occur as a type, but on the other hand, also act as an object and *have* a type. One refers to such expressions as *domains*. Their types are called *classes*. The contexts in CTT are not sets, but they are sequences of introductions. An introduction postulates the existence of a primitive object within a domain, and furnishes it with a (new) name.

Context construction starts with an initial context (which only contains the classes) that can be extended recursively. To postulate the existence of an object that is an inhabitant of a type T, the context is extended by a pair $[n : T]$, provided n is a new name, and provided the type T can indeed be shown to be a domain in the context C. In most systems this is accomplished by showing that that T is the inhabitant of a class, i.e., that $C \vdash [T : K]$, where K is a class. Classes are predefined in the initial context. The formalism, then, allows you to form contexts, and, given a context, to infer which expressions are welformed, and what their types are. In principle, that's what type theory is all about. Nevertheless, Constructive Type Theory is expressive enough to formalize and check complicated mathematical texts within it (cf. [6]). This is primarily due to the central concepts of ***propositions as types*** and ***dependent function types*** described below.

Propositions as Types: At first sight there may seem to be an unbridgeable gap between a type system and a logical system. Type systems seem to be concerned

with objects and classifications on objects, whereas the essential notion within a logical system is primarily that of a proof, together with that of a proposition. To bridge the gap, one could consider a proof as a mathematical object like any other, such as a number or a function. Consequently, a type system would concern itself with classifications of proofs, too. It turns out that proofs can be classified according to the propositions that they prove, i.e., a proposition can be regarded as a domain, containing all its proofs. Thus, within the system there exist domain-expressions that represent propositions. Proving a proposition, then, is finding an inhabitant of that proposition, i.e., constructing an object that has the desired proposition as a type. Thus proof-checking (even for higher order logic) is reduced to type-checking in a hierarchical type system, which is a completely mechanical task.

Dependent Function Types: If A and B are domains, we can form the abstraction expression $\Pi[x : A]B$, which also denotes a domain, the dependent product of B indexed over A. The intuition behind a dependent product is that it is a generalisation of the conventional function type $A \to B$. Indeed, we will find it convenient to write $A \to B$ instead of $\Pi[x : A]B$ if x does not occur in B. As an example of a case where x does occur in B, consider the function f that, given a natural number x, has as an image the null vector in R^x, i.e, $f : \Pi[x : N]R^x$. The domain of this function depends on the argument x , and the type of f cannot be represented in the conventional arrow notation. In our context, however, the principal use of dependent function types is in modelling propositions. For instance, once a predicate p on the type 'man' has been defined by an introduction like:

$$[p : man \to prop]$$

one can construct types like

$$\Pi[x : man](p\,x),$$

which, as we will see, can stand for the proposition that all men have property p.

Practical Use. As mentioned before, in practice the system has been used to formalize theories in such a way, that proofs within these texts can be checked mechanically. To translate a particular theory, first its basic entities are introduced, followed by the primary functions and predicates that play a role in the theory. Finally, *atomic proof objects* are added, which are inhabitants of the axioms relevant in the theory. These axioms are vital because only they provide some *content* to the predicates and functions that were introduced. All these introductions taken together form a context C that represents the bare theory. To prove that a certain theorem holds in this theory, it is formulated as a proposition P within the system. If an object Q can be constructed such that $C \vdash [Q : P]$, P has been shown to be a theorem. It will be evident that the object Q can be useful later when more elaborate proofs are to be constructed, that use P as a lemma.

1.2 Formalisation

We will now give a formal description of a particular CTT, the system proposed by Coquand.

An *expression* is one of the following:

- a *class*: this is an element of $\{kind, prop, type\}$.
- a *name*: any alphanumerical string that does not denote a class.
- a λ-abstraction: $\lambda[x : T]D$, T and D are expressions and x is a variable. The intuition, is, of course, that the expression denotes a function with formal argument x of type T, with a body D. We have the usual notion of free and bound occurrences in expressions.
- a Π-*abstraction*: $\Pi[x : T]D$, which denotes a dependent function type. Free and bound occurences are defined in complete analogy to the case for λ. If x does not occur free in D we will take the liberty of writing $T \rightarrow D$.
- an *application expression*: (ab), where a and b are expressions. The intuition is that the expression a denotes a function that is applied to the argument represented by the expression b. As usual, function application associates to the left, so we have $((ab)c) = (abc)$.

A *fragment* can be one of the following:

- a *pair*: $[E : T]$ where E is an expression and T is an expression. The intuition is that E is to be an object of type T.
- *a concatenation of fragments*: $F1 \otimes F2$, where F1 and F2 are fragments. The operator $\otimes$ is an associative non commutative concatenation operator. Accordingly, we leave out unnecessary parenthesis.

This completes the definition of the syntactical structure of the formalism.

On these structures the following functions and predicates are defined:

- *substitutions* are introduced as functions from expressions to expressions, and as functions from fragments to fragments: $[E/x]E2$ denotes the substitution of the expression E for all free occurences of x in the expression $E2$. If $[S]$ and $[S1]$ are substitutions, then $[S1 \circ S]$ denotes the consecutive substitution that performs $[S1]$ after $[S]$, accordingly, for any expression E we have $[S1]([S]E) = [S1 \circ S]E$. This notation for substitutions is extended over fragments as follows: $[S][X : T] = [[S]X : [S]T]$ and $[S](F1 \otimes F2) = [S]F1 \otimes [S]F2$.
- The predicate *class(E)* on expressions E is defined as follows:

  $class(E)$ iff $E \in \{kind, prop, type\}$

  Classes act as the types of domains.

- The binary predicate $\geq_\beta$ on expressions A and B is defined as follows:

  $A \geq_\beta B$, iff A can be reduced to B using β-reduction steps.

  β-reduction corresponds to the usual notion on λ-expressions, (though we demand that the type of the argument in any reduction pair is equal to the domain of the function).

- The binary predicate $A =_\beta B$ on expressions A and B is defined as follows:

  $A =_\beta B$ iff $\exists C(A \geq_\beta C \wedge B \geq_\beta C)$.

- The following predicates are defined which are central to the formalism: $\mathit{context}(F)$ for a fragment F (written $\{F\}$), and $\mathit{valid\_sequent}(F_1, F_2)$ for fragments F_1 and F_2 (written $F_1 \vdash F_2$). We define these predicates via inference rules which utilize the shorthand $\mathit{dom}(C, D, K) = \{C\} \wedge C \vdash [D : K] \wedge \mathit{class}(K)$. The intuition behind the *dom* predicate is that the expression D is a domain in a context C that is an inhabitant of the class K. The inference rules read as follows:

$$\frac{}{\{[\mathit{type} : \mathit{kind}] \otimes [\mathit{prop} : \mathit{kind}]\}} \text{ (init)}$$

The initial fragment, that introduces the classes, is a context. In this version of CTT, both *prop* and *type* are inhabitants of the class ***kind***.

$$\frac{\mathit{dom}(C, D, K) \wedge x \text{ does not occur free in } C}{\{C \otimes [x : D]\}} \text{ (intro)}$$

If it can be shown that the type of an expression D within a context C is a class, then D is a domain expression, and one may extend C with postulated inhabitants of D. Of course, the newly introduced object x should be given a fresh name.

$$\frac{\{C_1 \otimes [x : T] \otimes C_2\}}{C_1 \otimes [x : T] \otimes C_2 \vdash [x : T]} \text{ (selection)}$$

If a context C contains a pair $[x : T]$ then x has type T in the context C.

$$\frac{C \vdash R_1 \wedge C \vdash R_2}{C \vdash R_1 \otimes R_2} \text{ (collection)}$$

All fragments derivable in a common context can be concatenated.

$$\frac{C \otimes [x : T] \vdash [B : D]}{C \vdash [\lambda[x : T]B : \Pi[x : T]D]} \text{ (λ-abstraction)}$$

To find the type of a function, it suffices to find the type of the body in a context extended with a formal argument.

$$\frac{C \vdash [E : A] \wedge C \vdash [Q : \Pi[x : A]D]}{C \vdash [(QE) : [E/x]D]} \text{ (application)}$$

The type of an application expression is the range-type of the function, if the argument can be shown to 'fit' the function. If the function type is dependent however, one must make the appropriate substitutions. This rule is related to the modus ponens rule in CFOL.

$$\frac{dom(C \otimes [x : A], D, K)}{C \vdash [\Pi[x : A]D : K]} \text{ (Π-abstraction)}$$

The type of a Π expression over a domain is equal to the type of the domain.

$$\frac{C \vdash [E : D] \wedge dom(C, T, K) \wedge D =_\beta T}{C \vdash [E : T]} \text{ (type conversion)}$$

The type of a given expression may be exchanged for any other that is β-equal, provided that the replacing type can be shown to be a domain.

1.2.1 Some Properties

The rules for $\vdash$ given above can be used recursively to determine the type of an object in a given context. This is straightforward because the structure of the object at hand always determines which rule to apply. The only rule that could be problematic in this respect, is the type conversion rule. It can be proven however, that the system described above has the Church/Rosser property under β-reduction, and that it is strongly normalising, as well. This implies that every reduction on an object or a domain always terminates and yields a unique normal form, thereby guaranteeing the decidability of the equality of domains. Consequently, the predicates $\{\}$ and $\vdash$ are decidable (...and proof checking can be mechanized).

It has been proven that the system is substitutive, a property we will employ in later sections:

$$\frac{C_1 \otimes [x : A] \otimes C_2 \vdash [O : T] \ \wedge C_1 \vdash [E : A]}{C_1 \otimes [E/x]C_2 \vdash [E/x][O : T]} \text{ (substitution)}$$

Furthermore, the system is closed under (object-)reduction:

$$\frac{C \vdash [E : T] \wedge C \vdash [E_1 : D] \wedge E_1 \geq_\beta E}{T \geq_\beta D} \text{ (object closure)}$$

This last property means, among other things, that a proof for a proposition T, which can be reduced, will still prove essentially the same after reduction. It turns out that β-reduction on proofs correponds to cut elimination.

1.3 Texts and Contexts

A slight change in representation allows us to formulate a system that is intuitively more appealing. First we introduce the following notions:

- A fragment F is a ***valid extension*** to a context C, iff $\{C \otimes F\}$. Note, that if $[x_1 : D_1] \ldots [x_n : D_n]$ is a valid extension to C, it follows that none of the x_i do occur free in C.
- For any valid extension we define the notion of an ***assignment*** as follows: An assignment for a pair $P = [x : T]$ in C is a substitution $[E/x]$, where E is an expression. An assignment $[A]$ for a valid extension $(F_1 \otimes F_2)$ in C is $[A_2 \circ A_1]$ where $[A_1]$ is an assignment on F_1 in C and $[A_2]$ is an assignment on $[A_1]F_2$ in C.
- Let V be a valid extension on a context C. Let $[A]$ be an assignment for V in C. V is ***satisfiable*** on C through the assignment $[A]$ iff $C \vdash [A]V$ (Notation: $C[A]\vdash V$).

As a consequence of substitutivity it follows directly that

$$\frac{C_1 \otimes V \vdash G \ \wedge \ C_1[S]\vdash V}{C_1 \vdash [S]G} \text{ (satisfaction)}$$

These notions and properties play a role in the more general representation of fragments in terms of ***texts***, that we will develop below. Fragments are generalized to texts by the introduction of the connector ' $\Longrightarrow$ '. The underlying idea is that we have:

$$C \vdash V \Longrightarrow G \text{ iff } C \otimes V \vdash G.$$

Syntactically, a text is one of the following:

- a pair P
- an arrow $V \Longrightarrow G$ where V is a text and G is a text.

- a concatenation $G_1 \otimes G_2$, where G_1 and G_2 are texts.

We define a relation $\rightleftharpoons$ (*is congruent to*) between fragments and texts:

$$\frac{}{[E:T] \rightleftharpoons [E:T]} \text{ (ident)}$$

$$\frac{F_1 \rightleftharpoons G_1 \wedge F_2 \rightleftharpoons G_2}{F_1 \otimes F_2 \rightleftharpoons G_1 \otimes G_2} \text{ (conc)}$$

$$\frac{[(Qx):T] \rightleftharpoons G \wedge [x:A] \rightleftharpoons V}{[Q:\Pi[x:A]T] \rightleftharpoons (V \Longrightarrow G)} \text{ (unfold)}$$

$$\frac{[T:K] \rightleftharpoons G \wedge [x:A] \rightleftharpoons V}{[\Pi[x:A]T:K] \rightleftharpoons (V \Longrightarrow G)} \text{ (unfold2)}$$

$$\frac{F \rightleftharpoons (U \Longrightarrow (V \Longrightarrow G))}{F \rightleftharpoons ((U \otimes V) \Longrightarrow G)} \text{ (uncurry)}$$

$$\frac{F_1 \rightleftharpoons (V \Longrightarrow G_1) \wedge F_2 \rightleftharpoons (V \Longrightarrow G_2)}{F_1 \otimes F_2 \rightleftharpoons (V \Longrightarrow (G_1 \otimes G_2))} \text{ (distr)}$$

Given this congruency, we can write a text G instead of a fragment F. This means that for a given fragment there exist many notational variants, which are all equivalent, at least, if one can show that $F_1 \rightleftharpoons G \wedge F_2 \rightleftharpoons G$ implies $F_1 = F_2$. This is indeed the case, because the congruency is inversely related to a rewrite system that is strongly normalising. The proof rests on the absence of non conflatable critical pairs in the rewrite system. All predicates defined on fragments can now be extended to predicates over texts in a direct way. Deduction using texts is more intuitive. For instance, the following inference rules for texts can be derived from the old rules for fragments:

$$\frac{\{C \otimes U \otimes V\}}{C \otimes U \otimes V \vdash U} \text{ (select)}$$

$$\frac{C \otimes V \vdash G}{C \vdash V \Longrightarrow G} \text{ (assume)}$$

$$\frac{C[S\rangle V \wedge C \vdash V \Longrightarrow G}{C \vdash [S]G} \text{ (modus ponens)}$$

$$\frac{C \vdash G_1 \ \wedge \ C \vdash G_2}{C \vdash G_1 \otimes G_2} \text{ (collect)}$$

$$\frac{C \vdash [X : T] \ \wedge \ T \geq_\beta D}{C \vdash [X : D]} \text{ (convert-type)}$$

That these rules hold follows from proof theoretical arguments, which show that for any derived rule there would exist a derivation tree in the original system (modulo the congruence $\rightleftharpoons$). With them we have defined a formal system in natural deduction style, based on constructive type theory.

Before we proceed to the next section, where we will translate Discourse Representation Structures into texts, it is important to note some properties of arrows:

- In any arrow $V \Longrightarrow G$, the text V contains introductions of variables that originate from introductions of bound variables in Π expressions. Because bound variables are subject to α-conversion, one can rename these variables freely troughout the arrow. We call these variables the 'slots' of the arrow. Intuitively all these slots are variables that are 'introduced' in V.
- Let C be a context, and $C \vdash V \Longrightarrow G$ Let U be any valid extension to C. Now one can always rename the 'slots' of V in such a way that V is a valid extension to $C \otimes U$ (or to C itself).
- Let $[S]$ be a satisfying assignment for a text $V \Longrightarrow G$ on a context C. Let U be an arbritrary extension of C. Let $[S_1]$ be *any* satifying assignment for V on $C \otimes U$. Then it holds that $C \otimes U \vdash [S][S_1]G$. (This is a consequence of modus ponens, $[S]$ can be exchanged with $[S1]$ because the substitutions do not intersect.) Thus, a satisfying assignment for $V \Longrightarrow G$ on a context C insures that on any extension of C there exists a satisfying assignment for G if a satisfying assignment for V exists.

2 Discourse Representation

2.1 DRT

Within Discourse Representation Theory (DRT) a discourse is represented by a DRS (a Discourse Representation Structure). A DRS consists of two parts:

- a sequence of (discourse-)referents
- a sequence of entries

We will represent a sequence of referents by a list $(r_1 \ldots r_n)$. The empty list is written as $()$.

A DRS will be written as $(r_1 \ldots r_n) * E_1 \otimes E_2 \ldots E_n$, where $E_1 \ldots E_n$ are entries. An entry is either:

- an *atomic condition*, i.e., an n-ary predicate applied to a number of referents.
- a *complex condition*, written as $D_1 \Rightarrow D_2$.
- a *link* $[R = N]$, where R is a referent. and N is a name in the model.

A DRS is interpreted in relation to a model W. This model contains entities and predicates over these entities. The entities have names, and the predicates have predicate symbols. To keep things simple we will allow a DRS to refer directly to these entities and predicates, through the use of the names and the predicate symbols.

An assignment on a DRS $(r_1 \ldots r_n) * E_1 \ldots E_m$ is a conventional substitution, that substitutes a name in the model for each referent, that is, it assigns an entity in the model to each referent $r_1 \ldots r_n$. A DRS D is verifiable in relation to a model W if there exists an assignment $[A]$ from all the referents $r_1 \ldots r_n$ of D to the entities in W such that all the resulting entries in D are true. We call this assignment a verifying assignment of D in W.

Let $[A]$ be an assignment on D. An entry E' , where E' is $[A]E$ in the DRS D is true if one of the following holds:

- E' is an atomic condition and the proposition E' holds in W.
- E' is a link $[N' = N]$ and it is true in W that $N' = N$.
- E' is a complex condition $D_1 \Longrightarrow D_2$ and for any assignment $[S1]$ that verifies D_1 the DRS $[S1]D_2$ is verifiable in W.

A proof that this last condition is fullfilled must produce a verifying assignment $[S]$ for any $[S_1]$ that verifies D_1, i.e., an $[S]$ such that all the entries in $[S][S_1]D_2$ are true.

2.2 Translation of DRS to Text

The similarities between a DRS and a text are immediately transparent. To translate a DRS into a text, a relation *trans* between a DRS and a text is defined by the following rules:[3]

$$\frac{trans(R, A, U) \wedge trans(D, A, V)}{trans(R * D, A, U \otimes V)} \text{ (t-drs)}$$

$$\frac{trans(D_1, A, V_1) \wedge trans(D_2, A, V_2)}{trans(D_1 \otimes V_1, A, D_2 \otimes V_2)} \text{ (t-concat)}$$

[3]Where $append(A, B, C)$ means that C is the result of concatenating A and B.

$$\frac{reflist((r_1 \ldots r_n))}{trans((r_1 \ldots r_n), (), [r_1 : entity] \otimes \ldots [r_n : entity])} \text{ (t-ref-empty)}$$

$$\frac{reflist((r_1 \ldots r_n))}{trans((r_1 \ldots r_n), (a_1 \ldots a_n), [(r_1\, a_1 \ldots a_n) : entity] \otimes \ldots [(r_n\, a_1 \ldots a_n) : entity])} \text{ (t-ref-args)}$$

$$\frac{trans(D_1, (), V) \wedge trans(D_2, A_r, G) \wedge D_1 = A_1 * C \wedge append(A_0, A_1, A_r)}{trans(D_1 \Rightarrow D_2, A_0, V \Longrightarrow G)} \text{ (t-cond)}$$

$$\frac{proposition(P)}{trans(P, (), [p : P])} \text{ (t-prop-empty)}$$

$$\frac{proposition(P)}{trans(P, (a_1 \ldots a_n), [(p\, a_1 \ldots a_n) : P])} \text{ (t-prop-args)}$$

$$\frac{link([R = N])}{trans([R = N], (), [p : (eq\ R\ N)])} \text{ (t-link-empty)}$$

$$\frac{link([R = N])}{trans([R = N], (a_1 \ldots a_n), [(p\, a_1 \ldots a_n) : (eq\ R\ N)])} \text{ (t-link-args)}$$

Given this definition (which can almost be read as a PROLOG program), any proof of $trans(D, (), V)$, produces a text V that is a translation of the DRS D.[4]

Our mapping translates all referents to variables of type 'entity'. The reason for this lies, of course, in the somewhat impoverished (i.e., non-existent) typesystem of DRT, and is by no means required by the CTT formalism. Nor is the use of atomic types. In CTT, types like *(human male)*, *(human female)*, *animal* or *object* could be used in order to impose some taxonomy on the objects in the domain of discourse.

2.3 Semantics

We will investigate the question as to how far the presented translation preserves the underlying semantics. Let D be a DRS and W a model, and assume that D is verifiable on W. Now this means, that for all referents $(r_1 \ldots r_n)$ of D there exist substitutions, such that all the entries are true on W. Let V be a text such that $trans(D, (), V)$. By analogy, a text V is verifiable on a context C if there exists an assignment $[A]$ for V on C with $C[A]\text{–}\ V$. But how can a context C be found, that can serve as a basis for the judgement of the text V? This is not as tricky as it seems,

[4]Note, that, strictly speaking, we can only translate a DRS that is not empty. Of course, one can introduce the notion of an empty fragment '[]' that serves as the identity element of the concatenation operator '⊗' to mend this.

because one can also pose a similar question for DRT. How can one find a suitable model W in relation to which D should be judged? Even if one believes that any discourse should relate to the actual world, there is no sensible way of constructing a model for it: No one has complete knowledge of the world, and there is no hope of ever knowing it. The answer is, then, that any model W will be constructed by an agent, and therefore it will reflect the knowledge, or more precisely, the *beliefs* of its constructor. If, as in the case of DRT, verification is based on a complete model, the poor constructor is also obliged to add fiction for those parts of the world he does not know. (This resembles the way in which the artists filled the blank spaces in medieval maps.) In CTT the situation is rather different. Contexts do not pretend to be a model of the world, they are *inherently* partial and therefore suited perfectly to represent some agent's beliefs about the world. Thus, to interpret utterances within the framework of CTT, the corresponding (CTT-)text will be judged using a context C_j of an agent J. Accordingly, the resulting judgement will not be universal nor absolute, but it will precisely reflect the judgement of that particular agent.

2.3.1 Simulating 'classical' semantics

In order to reproduce a 'classical' semantics, like that of DRT, one has to assume the existence of an 'omniscient' agent Ω, whose knowledge state C_ω implicitly contains all the knowledge of the 'real' world W.

The agent Ω needs to have proofs for all first order statements that hold in W, and disproofs for all the statements that do not hold. The context C_ω must contain introductions for all entities in W, and closure axioms that exclude the existence of other entities. Equality could be handled as an ordinary predicate. Finally axioms must be provided that allow construction of all 'first order' functions within the context C_ω.
To show that this leads to equivalent semantic notions we need to prove that:

$$\frac{\exists[A](C_\omega[A_\omega]\vdash V) \ \wedge \ trans(D,(),V)}{\exists[A'](verif(W,[A'],D))} \ \text{(back)}$$

and

$$\frac{\exists[A'](verif(W,[A'],D)) \wedge trans(D,(),V)}{\exists[A_\omega](C_\omega[A_\omega]\text{–}\ V)} \ \text{(forth)}$$

where the predicate $verif(W,[A'],D)$ means that $[A']$ is a verifying assignment for D in the model W. For our present purposes a short sketch of the necessary elements of such a proof will suffice.

One should note right away that the assignment $[A_\omega]$ is in many ways more explicit than its counterpart $[A']$: $[A_\omega]$ does not only assign objects to entities, but also contains formal proofs that the substituted entities satisfy their requirements, i.e., that $[A']$ is, indeed, a verifying assignment.

So it comes as no surprise, that it is easy to construct $[A']$ if $[A_\omega]$ is given: $[A_\omega]$ assigns to its variables either entities, or proof objects, or functions that yield entities, or functions that yield proof objects. By simply leaving out all the proof objects and the functions in $[A_\omega]$ we regain the assignment $[A']$. The proof objects in $[A_\omega]$ show that the assignment $[A']$ satisfies the properties it must satisfy. The functions can be used to demonstrate that satisfying assignments exist for the consequents of conditions, given arbritrary satisfying assignments for the antecedents of the conditions.

But to establish equivalence we must also show that we can construct $[A_\omega]$ if $[A']$ is given. This, of course, is more difficult, as the assignment $[A']$ in itself does not suffice to show that it satisfies D in W. To derive $[A_\omega]$ anyway, we have to take into account not only $[A']$, but also the ***truth conditions*** on $[A']$ (cf. p. 10), that is, we have to make explicit, and, so to say, hard-code into $[A_\omega]$ the relevant properties of the metalangage of $[A']$. One of these properties is the requirement that the substitution $[A']$ must be such that all resulting entries in D are true in W. As far as propositions and links are concerned, this property of $[A']$ ensures that the required proof objects that we need to extend $[A']$ with in order to construct $[A_\omega]$ are indeed constructable within C_ω. The situation is more complicated in the case of complex conditions $D1 \Longrightarrow D2$. Here the existence of satisfying substitutions for the referents of $D2$ is required for any satisfying assignment to the referents of $D1$. Now this boils down to the existence of functions on the model W that produce a satisfying assignment for any referent in $D2$ given a satisfying assignment for $D1$. Remember that we imposed on C_ω the requirement that all first order functions that exist on W can be constructed in C_ω. So proofs for the properties of these functions can be constructed, too, and $[A']$ can be extended with the required substitutions.

2.3.2 The constructive approach

After this short digression into the realms of mythology let us return to the real life issues: In real life, there is no principal reason why the knowledge of some agent should be complete, or even accurate. On the contrary, all utterances are judged by agents that have only partial, and often even incorrect knowledge of a state of affairs. It is of crucial importance that such situations can be represented and understood formally. Thus the communicative reflections of a common mortal seem to be a lot more interesting to investigate than the godlike verdicts of Ω. Therefore we will now introduce a common mortal agent J, and study its judgements in a number of examples. These examples suggest that interesting situations, which occur in real life, but cannot easily be handled in a classical approach, receive a quite natural treatment in CTT. Among these are cases where an agent is provided with 'new' information, and utterances involving scope ambiguity.

We assume that the world knowledge of the agent J is described by a context C_j. It contains the basic mathematical and logical knowledge that the agent uses, it contains all entities in the world, whose existence it assumes, as well as introductions

and axiomatisations of all the predicates and functions on these entities that the person conceives of. As an example, take the following context[5]:

$[type : kind]$
$[prop : kind]$
The context of any agent must be an extension of the initial context.

$[entity : type]$
This agent only introduces one type, for the sake of simplicity. Note, that, as a consequence, it has immediately created all the associated Curry types.

$[x : entity][y : entity] \Longrightarrow [(eq\, x\, y) : prop]$
The agent knows that equality is a relation within the type.

$[p : entity \rightarrow prop] \otimes [x : entity] \otimes [y : entity] \otimes [g : (p\, x)] \otimes [z : (eq\, x\, y)]$
$\Longrightarrow [(leibniz\, p\, x\, y\, z\, g) : (p\, y)]$
It gives meaning to equality using leibnitz rule.

$[x : entity] \Longrightarrow [(refl\, x) : (eq\, x\, x)]$
and by postulating reflexivity of equality.

$[contradiction : prop]$
The agent conceives of something called a contradiction.

$[x : contradiction][p : prop] \Longrightarrow [(x\, p) : p]$
It also knows that a proof of the contradiction would allow you to prove anything. The contradiction is very important because it enables the agent to formalize negative knowledge. For instance, to formalize that certain entities are not equal, it can define a transitive relation that is not reflexive.

Of course, the agent also has knowledge (beliefs) about the state of affairs in the world:

$\vdots$

$[x : entity] \Longrightarrow [(farmer\, x) : prop]$
$[x : entity] \Longrightarrow [(donkey\, x) : prop]$
$[x : entity] \otimes [y : entity] \Longrightarrow [(hits\, x\, y) : prop]$
$[x : entity] \otimes [y : entity] \Longrightarrow [(owns\, x\, y) : prop]$

$[pedro : entity]$
$[jerry : entity]$
$[f1 : (farmer\, pedro)]$
$[f2 : (donkey\, jerry)]$
$[f3 : (owns\, pedro\, jerry)]$

[5]For lay-out reasons we have replaced the outer occurences of the operator '$\otimes$' within this context by vertical spacing

$[x : entity] \otimes [y : entity] \otimes [p1 : (farmer\ x)] \otimes [p2 : (donkey\ y)][p3 : (owns\ x\ y)]$
$\Longrightarrow [(r1\ x\ y\ p1\ p2\ p3) : (hits\ x\ y)]$
Every farmer who owns a donkey, hits it.

$[x : entity] \otimes [p1 : (owns\ x\ pedro)] \Longrightarrow [(r2\ x\ p1) : contradiction]$
No one owns pedro.

$\vdots$

&c.

Given this context, suppose our agent is confronted with the following utterances:

- ***A farmer owns a donkey. He hits it.***

It will be evident, that under a translation process analogous to that of DRT, the following text V will result:[6]

$[f : entity] \otimes [p : (farmer\ f)] \otimes [d : entity] \otimes [p2 : (donkey\ d)]$
$\otimes [p1 : (owns\ f\ d)] \otimes [p3 : (hits\ f\ d)]$

The agent should judge this text. V is a valid extension to C_j , i.e., we have $\{C_j \otimes V\}$. It is also a satisfiable extension, which means that the utterance is judged to be true. The satisfying assignment $[A_j]$ with $Cj[A_j]\vdash V$ reads as follows:

$$A_j = [(r1\ pedro\ jerry\ f1\ f2\ f3)/p3 \circ f3/p1 \circ f2/p2 \circ jerry/d \circ f1/p \circ pedro/f]$$

- *jerry hits pedro.*

This utterance is translated into:

$[z : (hits\ jerry\ pedro)]$.

Again, it is a valid extension of the context C_j. But, in this case, no satisfying assignment exists, the sentence does not follow from the context C_j. But falsehood cannot be proven either, and nothing prevents our agent J to extend its context C_j with this new and exciting information, if it chooses to consider it valid. The advantages of the constructive, 'partial' approach to cases like this is obvious.

- *every farmer owns jerry.*

This is translated into the text V_2:

$[x : entity] \otimes [p : (farmer\ x)] \Longrightarrow [(q\ x\ p) : (owns\ x\ jerry)]$

[6] A direct translation algorithm into CTT which preserves ther DRT restrictions on anaphora resolution, but allows for an elegant treatment of a much wider range of linguistic phenomena than the original DRT fragment in [7], will be given in [10].

Again a valid extension, but even if there are no entities within C_j that are farmers except pedro, the utterance is still contentious. No satisfying assignment for V_2 can be found. This is a consequence of the fact that the agent is neither omniscient, nor does it believe that it knows all farmers that exist. Consequently, it does not have a way to verify the statement. If, however, the agent *does* believe that it knows all the farmers, this conviction should be expressed within the context, for instance by a clause like

$$[p : entity \rightarrow prop] \otimes [z : (p\, pedro)] \otimes [f : entity] \otimes [q2 : (farmer\, f)] \\ \Longrightarrow [(farmerclosure\, p\, z\, f\, q2) : (p\, f)]$$

If we incorporate this axiom[7] into C_j we can indeed reach the relevant conclusion, using

$$[A_j] = [(farmerclosure\, \lambda[e : entity](owns\, e\, jerry) f3)/q]$$

$$[x : entity] \otimes [p : (farmer\, x)] \\ \Longrightarrow [((farmerclosure\, \lambda[e : entity](owns\, e\, jerry) f3)\, x\, p) : (owns\, x\, jerry)]$$

- *every farmer owns something.*

with the translation V_3

$$[x : entity] \otimes [p : (farmer\, x)] \Longrightarrow [(d\, x\, p) : entity] \otimes [(q\, x\, p) : (owns\, x\, (d\, x\, p))]$$

This example, though qua 'truth conditions' comparable to the previous one, has an interesting additional hitch: it involves scope ambiguities. Again, our agent does not assume farmer-closure. Thus, there is no satisfying assignment for V_3. If the agent chooses to believe the statement, it will extend its context with V_3. But, when doing so, no decision has to be made about the scopes of the quantifiers. Even the introduction of new farmers doesn't cause any trouble. If, by some coincidence, it turns out that all farmers own the same thing, all that happens is that information about the function d is added to the context, like, e.g. $\lambda[x_1 : entity]\, \lambda[p_1 : (farmer\, x_1)]\, jerry$, showing that this function is, in fact, a constant.

Finally, let us look at an example which gives rise to quite a different situation:

- *jerry owns pedro.*

translated into:

$$[z : (owns\, jerry\, pedro)].$$

This translation, too, is a valid extension of C_j. But now there is not only no satisfying assignment, but the extension even leads to an inconsistent knowledge state, i.e., within the extended context, the contradiction is provable:

$$C_j \otimes [z : (owns\, jerry\, pedro)] \vdash [(r2\, jerry\, z) : contradiction]$$

[7] Note that this axiom is much stronger than the (possible) accomodation of C_j by adding V_2.

So, in the context C_j, the statement is judged to be false and will be rejected.

It is worth noting, however, that rejection is not the only possible way to resolve this dilemma. If our agent has reason to believe the speaker, it might want to update its context in a non-monotonic way in order to add z without ending up with inconsistent knowledge. Although the formalism presented here does not cover non-monotonic belief maintenance, the presence of explicit proof objects in CTT provides us—in clear contrast to model theoretic approaches—with a fairly straightforward handle on such problems: It allows us to identify the 'villain' causing the inconsistency easily. In the case at hand, the problematic proof is $(r2\, jerry\, z)$. To accept z, our agent has to reject either $r2$ or $jerry$. If $jerry$ is rejected, anything that contains $jerry$ has to be removed, too—including z, since the existence of $jerry$ is a presupposition of the assertion $(owns\, jerry\, pedro)$. So this choice is ruled out. Therefore, if agent J insists on z, it is forced to reject $r2$. In cases where the resulting context still contains contradictions, this procedure can be repeated. We reserve the details of this way of updating beliefs for a subsequent paper.

3 Conclusions

We have shown that any DRS can be translated into a fragment in CTT. Judgements on such fragments can be made with respect to CTT contexts. The deductive system that lies at the roots of these judgements is very powerful, and the language is very expressive. As these contexts may represent the partial knowledge state of an agent, we can model how one agent judges the utterance of another, independent of any notion of a complete or absolute truth. This could be considered a first step towards a theory of communication.

3.1 Future directions

It seems reasonable to assume that a theory of discourse processing could profit from the expressivity of CTT in many ways. Apart from the fact that CTT is able to handle higher order expressions, as in the induction example, the types that CTT offers can be used to structure the domain of discourse in a formally sound way. Moreover, there are several natural extensions of CTT (e.g., definitions & dependent sums) which facilitate the task of constructing linguistically reasonable ('quasi-compositional') translations from natural language to CTT. We will expand on some of these issues in [10]. Finally, we would like to note that it is possible:

- To translate a limited kernel of natural language into the formalism.
- To have an automatic deduction facility for the formalism in question [5].
- To generate natural language from fragments in the formal language [11].

Thus, one might be inclined to think that a primitive reasoning agent, able to interact within a limited natural language fragment, can be constructed along these lines.

References

[1] Ahn, R. (1985): 'Automath as an Extension to PROLOG.' Technical Note 173/85, Philips Research Laboratories, Eindhoven.

[2] De Bruijn, N.G. (1980): 'A Survey of the Project Automath.' In: Seldin & Hindley (eds.): *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalisms.* Academic Press.

[3] Coquand, T. (1985): *Une théory des Constructions.* Thèse de troisième cycle, Université de Paris VII.

[4] Curry, H.B. & R. Feys (1958): *Combinatory Logic, Vol. 1.* Dordrecht: North Holland.

[5] Helmink, L. & R. Ahn (1988): 'Goal Oriented Proof Construction in Type Theory.' To appear in *Journal of Automated Reasoning.*

[6] Jutting, L.S. (1976): *A Translation of Landau's* Grundlagen *in AUTOMATH.* PhD. Diss., Eindhoven University.

[7] Kamp, H. (1981): 'A Theory of Truth and Semantic Representation.' In: J.A.G. Groenendijk et al. (eds.): *Formal Methods in the Study of Natural Language.* Amsterdam.

[8] Kamp, H. (1985): 'Situations in Discourse without Time or Questions.' Ms., Univ. of Texas, Austin.

[9] Kolb, H.-P. (1987): 'Diskursrepräsentationstheorie und Deduktion.' *Linguistische Berichte* 110.

[10] Kolb, H.-P. & R. Ahn (in preparation): 'Towards a Constructive Theory of Communication.'

[11] Mäenpää, P. & A. Ranta (1989):'An Implementation of Intuitionistic Categorial Grammar.' This Volume.

[12] Martin-Löf, P. (1984): *Intuitionistic Type Theory.* Napoli: Bibliopolis.

[13] Mönnich, U. (1985): *Untersuchungen zu einer konstruktiven Semantik für ein Fragment des Englischen.* Habilitationsschrift, Tübingen University.

ITK: P.O. BOX 90153 5000 LE TILBURG THE NETHERLANDS